IN THE CLAIMS

Please amend the claims as follows:

1. (Original) A toolkit for developing user-interfaces for a system administration program, comprising:

a server-side application-programming interface (API), comprising a Task-registry file, wherein the Task-registry file comprises one or more Task groups; and

a client-side API, comprising a product-specific properties file, wherein the product-specific properties file is customizable by a developer and the client-side API is callable by developer-supplied code to create a graphical user interface for a specific product.

2. (Original) The toolkit of claim 1 wherein the properties file comprises:
a title; and
a product-specific list of page links.

3. (Original) The toolkit of claim 1, wherein the client-side API creates a Task-manager window comprising:
a customizable title.

4. (Original) The toolkit of claim 1, wherein the client-side API creates a Task-manager window comprising:
a customizable table of contents.

5. (Original) The toolkit of claim 1, wherein the client-side API creates a Task-manager window comprising:
a customizable display area.

6. (Original) The toolkit of claim 1, wherein the client-side API creates a Task-manager window comprising:
a customizable button bar.

7. (Original) The toolkit of claim 1, wherein the client-side API creates a Task-manager window comprising:
a display area.

8. (Original) The toolkit of claim 7, wherein the client-side API creates a Task-manager window comprising:
one of a text page, a Task-list page, and a class page.

9. (Original) The toolkit of claim 8, wherein the Task-list page comprises:
a list of Tasks that are related.

10. (Original) The toolkit of claim 9, wherein the Tasks are related by a type of object on which they operate.

11. (Original) The toolkit of claim 1, wherein the product-specific properties file comprises a ordered set of button tags, wherein each button has a name and a target class to be launched when the button is activated.

12. (Original) The toolkit of claim 1, further comprising a resource file, wherein the resource file is customizable by the developer.

13. (Original) The toolkit of claim 12, wherein the developer-supplied code uses the server-side API to create an Item, wherein the Item represents a system entity to be administered.

14. (Original) The toolkit of claim 12, wherein the client-side API and the resource file can be used by the developer-supplied code to create an ItemView.

15. (Original) The toolkit of claim 12, wherein the server-side API can be used by the developer-supplied code to create a Category, wherein the Category represents a collection of monitored Items of a specific type.

16. (Original) The toolkit of claim 12, wherein the client-side API and resource file can be used by the developer-supplied code to create a CategoryView.

17. (Original) The toolkit of claim 12, wherein the client-side API and resource file can be used by the developer-supplied code to create a TreeView.

18. (Original) The toolkit of claim 12, wherein the client-side API and resource file can be used by the developer-supplied code to create a Task.

19. (Original) The toolkit of claim 12, wherein the client-side API and resource file can be used by the developer-supplied code to create a ResultView.

20. (Original) The toolkit of claim 1, wherein the client-side API provides RichTextComponents that comprise glossary links and task launchers.

21. (Original) The toolkit of claim 1, wherein the client-side API provides blocking dialogs.

22. (Original) The toolkit of claim 14, wherein the ItemView is launched from a ResultView.

23. (Original) The toolkit of claim 22, wherein the ResultView displays an affected Item's name.

24. (Original) The toolkit of claim 23, wherein the name is updated when the Item changes.

25. (Original) The toolkit of claim 15, wherein the client-side API provides an Item Table, wherein the Item Table displays all Items in the Category in table form.

26. (Original) The toolkit of claim 1, wherein the client-side API provides an ItemFinder, wherein the ItemFinder populates itself with names of Items in a Category.

27. (Original) The toolkit of claim 1, wherein the client-side API renders icons dynamically from a vector-based icon description.

28. (Original) The toolkit of claim 17, wherein the TreeView displays a hierarchical view of Items in cascading Categories.

29. (Original) The toolkit of claim 1, wherein the client-side API provides icons that blink to reflect the state of an object.

30. (Original) The toolkit of claim 1, wherein the client-side API provides a splash screen, wherein the splash screen is displayed after an application is executed and before the application window is ready.

31. (Original) A signal-bearing media for developing user-interfaces for a system administration program, wherein the signal-bearing media comprises instructions and data, which when read and executed by a computer comprise:
a server-side application-programming interface (API), comprising a Task-registry file, wherein the Task-registry file comprises one or more Task groups; and
a client-side API, comprising a product-specific properties file, wherein the product-specific properties file is customizable by a developer and the client-side API is callable by developer-supplied code to create a graphical user interface for a specific product.

32. (Original) The signal-bearing media of claim 31, wherein the properties file comprises
a title; and
a product-specific list of page links.

33. (Original) The signal-bearing media of claim 31, wherein the client-side API creates a Task-manager window comprising:
a customizable title.

34. (Original) The signal-bearing media of claim 31, wherein the client-side API creates a Task-manager window comprising:
a customizable table of contents.

35. (Original) The signal-bearing media of claim 31, wherein the client-side API creates a Task-manager window comprising:
a customizable display area.

36. (Original) The signal-bearing media of claim 31, wherein the client-side API creates a Task-manager window comprising:
a customizable button bar.

37. (Original) The signal-bearing media of claim 31, wherein the client-side API creates a Task-manager window comprising:
a display area.

38. (Original) The signal-bearing media of claim 37, wherein the display area comprises:
one of a text page, a Task-list page, and a class page.

39. (Original) The signal-bearing media of claim 38, wherein the Task-list page comprises:
a list of Tasks that are related.

40. (Original) The signal-bearing media of claim 39, wherein the Tasks are related by a type of object on which they operate.

41. (Original) The signal-bearing media of claim 31, wherein the product-specific properties file comprises a ordered set of button tags, wherein each button has a name and a target class to be launched when the button is activated.

42. (Original) The signal-bearing media of claim 31, further comprising a resource file, wherein the resource file is customizable by the developer.

43. (Original) The signal-bearing media of claim 31, wherein the server-side API can be used by the developer-supplied code to create an Item, wherein the Item represents a system entity to be administered.

44. (Original) The signal-bearing media of claim 42, wherein the client-side API and resource file can be used by the developer-supplied code to create an ItemView.

45. (Original) The signal-bearing media of claim 31, wherein the server-side API can be used by the developer-supplied code to create a Category, wherein the Category represents a collection of monitored Items of a specific type.

46. (Original) The signal-bearing media of claim 42, wherein the client-side API and resource file can be used by the developer-supplied code to create a CategoryView.

47. (Original) The signal-bearing media of claim 42, wherein the client-side API and resource file can be used by the developer-supplied code create a TreeView.

48. (Original) The signal-bearing media of claim 42, wherein the client-side API and resource file can be used by the developer-supplied code to create a Task.

49. (Original) Thc signal-bearing media of claim 42, wherein the client-side API and resource file can be used by the developer-supplied code to create an ResultView.

50. (Original) The signal-bearing media of claim 31, wherein the client-side API provides RichTextComponents that comprise glossary links and task launchers.

51. (Original) The signal-bearing media of claim 31, wherein the client-side API provides blocking dialogs.

52. (Original) The signal-bearing media of claim 44, wherein the ItemView is launched from a ResultView.

53. (Original) The signal-bearing media of claim 52, wherein the ResultView displays an affected Item's name.

54. (Original) The signal-bearing media of claim 53, wherein the name is updated when the Item changes.

55. (Original) The signal-bearing media of claim 45, wherein the client-side API provides an Item Table, wherein the Item Table displays all Items in the Category in table form.

56. (Original) The signal-bearing media of claim 31, wherein the client-side API provides an ItemFinder, wherein the ItemFinder populates itself with names of Items in a Category.

57. (Original) The signal-bearing media of claim 31, wherein the client-side API renders icons dynamically from a vector-based icon description.

58. (Original) The signal-bearing media of claim 47, wherein the TreeView displays a hierarchical view of Items in cascading Categories.

59. (Original) The signal-bearing media of claim 31, wherein the client-side API provides icons that blink to reflect the state of an object.

60. (Original) The signal-bearing media of claim 31, wherein the client-side API provides a splash screen, wherein the splash screen is displayed after an application is executed and before the application window is ready.